Requirements Module: Writing Requirements

Space Systems Engineering, version 1.0

*Good Requirements Are **SMART***

- **S**pecific -
  - It must address only one aspect of the system design or performance
  - It must be expressed in terms of the need (what and how well), not the solution (how).
- **M**easurable -
  - Performance is expressed objectively and quantitatively
  - E.g., an exact pointing requirement (in degrees) can be tested thus verified prior to launch.
- **A**chievable -
  - It must be technically achievable at costs considered affordable
  - E.g., JWST early designs specified an aperture requirement eventually descoped due to technical issues with deployment.
- **R**elevant -
  - It must be appropriate for the level being specified
  - E.g., requirement on the solar cells should not be designated at the spacecraft level.
- **T**raceable -
  - Lower level requirements (children) must clearly flow from and support higher level requirements (parents).
  - Requirements without a parent are referred to as orphans, and need to be assessed for necessity of inclusion.

Rules for Writing Good Requirements

- Requirements have mandatory characteristics:
  - *Needed*
  - *Verifiable*
  - *Attainable: technically, cost, schedule*
- Each requirement should
  - *Express one thought*
  - *Be concise and simple*
  - *Be stated positively*
  - *Be grammatically correct; free of typos and misspellings*
  - *Be understood only one way; they are unambiguous*
  - *Use consistent terminology to refer to the system/product and its lower level entities*
  - *Comply with the project's template and style rules*

Distinguish Between 'Desirements' and Requirements

- *Desirement* – something that would be nice to have but is not mandatory to accomplish the mission
- Requirement - something that must be done for the mission to be successful
- Customers and stakeholders often blur the distinction between what is necessary and what would be nice to have

- Develop the problem statement independently of the solution
- Customers, stakeholders, design engineers and even systems engineers often have solution biases – Desirement space
- *Be Careful* - Customers or stakeholders may have a good idea, but early implementation solutions may prove impossible, impractical or sub-optimal when looked at in detail - capture what is necessary for mission success and *then* develop solutions to meet those needs.

More Rules for Writing Good Requirements

What a requirement must state:

- **WHO** is responsible
- **WHAT** shall be done
- Or **HOW WELL** something shall be done
- Or under what **CONSTRAINTS** something shall be done

Requirement format: **"WHO shall WHAT"**

- Uses active not passive voice

Example product requirements:

- The system shall operate at a power level of…
- The software shall acquire data from the…
- The structure shall withstand loads of…
- The hardware shall have a mass of…

Use the correct terms:

- *Requirements* are binding - **Shall**
- *Facts* or Declaration of purpose - **Will**
- *Goals* are non-mandatory provisions - **Should**

- Do NOT use "Must"

Goodness Checklist: Is this Requirement…

- Free of ambiguous terms?
  - Examples: as appropriate, etc., and/or, support, but not limited to, be able to, be capable of
- Free of indefinite pronouns?
  - Examples: this, these
- Free of unverifiable terms?
  - *Examples: flexible, user-friendly, robust, light-weight, maximize, adequate, small, portable, easily - other "ly" words and other "ize" words*
- *Free of implementation?*
  - *Requirement should state WHAT is needed, NOT HOW to provide it, i.e., state the problem not the solution.*
- *Necessary?*
  - *Ask "Why do you need the requirement?"; the answer may lead you to the real requirement.*
- *Free of descriptions of operations?*
  - *To distinguish between operations and requirements, ask "Does the developer have control over this?" "Is this a need the product must satisfy or an activity involving the product?"*
- *Free of TBDs (To Be Determined)?*
  - *Use a best estimate and a TBR (To Be Resolved) with rationale when possible.*

Example Requirements: Good or Bad?

- **The aircraft shall have three engines (initial DC-3 requirement).**
  - The aircraft shall meet the operation requirements with a single engine out.
- **The lunar lander shall include an airlock.**
  - The lunar lander shall provide the capability for crew to ingress/egress while maintaining pressurization.
- **The crew shall have the capability to perform extra-vehicular activities (EVAs).**
  - The vehicle shall allow extra-vehicular activities during operations.
- **The spacecraft shall maximize lifetime.**
  - The spacecraft shall have a lifetime of at least three years.
- **The software shall display data in a user-friendly fashion.**
  - The software shall display data as described in ICD 2345 Table 3.1.

Rationale Captures the Motivation and Assumptions of a Requirement

The rationale of each requirement defines

- Why a requirement is needed
- What assumptions were made
- What design effort drove the requirement
- Other data that will be needed to maintain the requirement over time

Example

- Requirement: "The truck shall have a height of no more than 14 feet."
- Rationale: 99% of all US interstate highway overpasses have a 14 foot or greater clearance. (Assumptions: The truck will be used primarily on US interstate highways for long-haul freight in the US.)

Requirements Validation

- Requirements validation asks three questions:
  - *Do we have the correct problem?*
  - *Do our requirements capture this problem?*
  - *Are our requirements SMART?*
- Performed to *assure* the system requirements *set* is complete, consistent, and each requirement is achievable and verifiable.
- Performed by subject matter experts, the system performing organization and the system authorized customer.
- Does the requirement set completely address and accomplish the mission?
- Are the requirements consistent with the established system boundaries and constraints?
- When does requirement validation take place?
  - *Usually before the System Requirements Review (SRR)*

Requirements Verification and Requirements Validation

- Requirement verification is establishing confidence that the requirement has been met.
  - Requirements are verified by test, demonstration, analysis or inspection.
  - Test is the most common method of verification and the technique that provides the most confidence that the system will indeed work as intended in its anticipated environment.
- Requirement validation is a process of ensuring that :
  1. the *set* of requirements is correct, complete, and consistent,
  2. a model can be created that satisfies the requirements, and
  3. a real-world solution can be built and tested to prove that it satisfies the requirements.
- If a systems engineer discovers that the customer has required a perpetual-motion machine, the project should be stopped.

Module Summary: Writing Requirements

- Good requirements Are SMART: **S**pecific, **M**easurable, **A**chievable, **R**elevant and **T**raceable
- It is good practice to capture the rationale and preliminary technique for verification when writing requirements.
- Requirement validation is a process of ensuring that: the *set* of requirements is correct, complete, and consistent.
- Since the cost of reconciling undefined requirements grows as the project matures, undefined (TBD) and estimated (TBR) requirements should be resolved as early as possible.